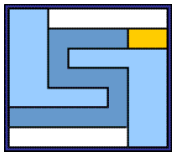
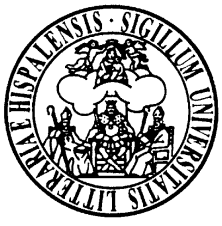
DATA MINING

Francisco J. Ferrer
Jesús Aguilar
Joaquín Peña

Departamento de Lenguajes y Sistemas Informáticos
Facultad de Informática
Universidad de Sevilla

INFORME TÉCNICO: LSI-2000-08
Julio, 2000





INDICE GENERAL

Introducción

Según [1] se estima que cada 20 meses se duplica la información en todo el mundo. La *Sociedad de la Información* destina gran cantidad de recursos en adquirir, almacenar y procesar enormes cantidades de información de muy diversa índole (financiera, comercial, industrial, científica, publicitaria, etc.) y todo ello a un ritmo que se acelera diariamente debido al constante desarrollo tecnológico donde *Internet*, base de datos dinámica y creciente llamada a ser la principal fuente de extracción de información, es sólo el pico visible del iceberg.

Hoy en día las grandes organizaciones se administran con el apoyo de sistemas de gestión que manejan terabytes de información, una información que oculta *conocimiento* en forma de reglas, asociaciones, excepciones o patrones, todos ellos de gran valor en la toma de decisiones de negocio. Frente a las respuestas proporcionadas por los tradicionales sistemas de gestión de bases de datos (*"¿Cuáles fueron las ventas en el tercer trimestre en la región norte?"*) aparecieron las herramientas de análisis multidimensional (también llamadas OLAP, On-Line Analytical Processing) que permiten realizar consultas mucho mas complejas (*"¿Qué ventas se preveen en el tercer trimestre del año próximo en la región norte?"*) y donde el tiempo aparece como una de las variables principales. Sin embargo, en ambos casos los valores obtenidos siguen derivando de los datos existentes.

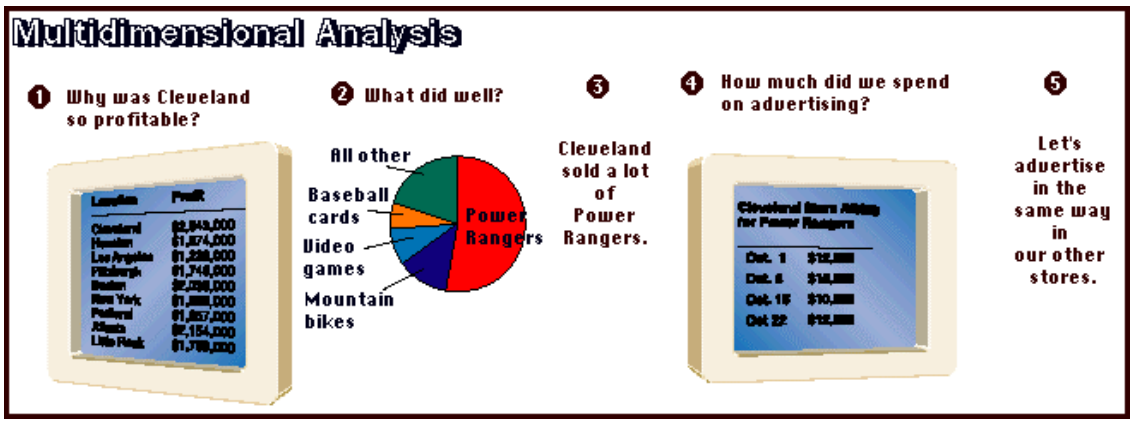


Figura 1. Diagrama del proceso OLAP.

El fuerte análisis estadístico sobre el que operan estas últimas consultas proporciona métodos para la búsqueda de correlaciones y dependencias entre los datos. Y mediante herramientas de visualización (a veces *una imagen dice más que mil palabras*) se identifican fácilmente tendencias en el tiempo entre varias medidas de un fenómeno. Pero no parece nada claro que las habilidades de un experto puedan con igual eficiencia analizar trillones de datos almacenados en soporte informático e inducir patrones desconocidos, en un momento en el que la diferencia entre la cantidad de datos manejados y el conocimiento que se tiene de los mismos aumenta continuamente. El interés está en *"¿Cómo aumentar las ventas en el tercer trimestre en la región norte?".*

A finales de la década de los 80, la creciente necesidad de automatizar todo este proceso inductivo abre una línea de investigación para el análisis inteligente de datos. De las similitudes entre buscar valiosa información de negocio en grandes bases de datos y minar una montaña para encontrar una veta de metales valiosos, el conjunto de métodos matemáticos y técnicas software para análisis inteligente de datos y búsqueda de regularidades y tendencias en los mismos, aplicados de forma iterativa e interactiva, se denominaron técnicas de **Minería de Datos** o ***Data Mining***.

KDD y *Data Mining*

Desde hace más de dos décadas se vienen desarrollando y utilizando complejos algoritmos para la extracción de patrones útiles en grandes conjuntos de datos. Unos de los pioneros en ello fue la administración de hacienda estadounidense que, mediante lógica difusa, redes neuronales y reconocimiento de patrones, ha intentado detectar fraudes en la declaración y evasión de impuestos [2].

Sin embargo durante todo este tiempo, las técnicas aplicadas fueron en su mayor parte dominio de las administraciones públicas. El elevado coste asociado y los enormes requisitos de almacenamiento, tiempo y memoria limitaron en gran medida el enorme campo de aplicación actual. Esta escasa difusión generó diversas nombres para una misma disciplina: *knowledge extraction, information descovery, information harvesting, data archaeology, siftware, data dredging, data pattern processing* y ***Data Mining***. Hoy en día el hardware ha dejado de ser el problema.

Los grandes avances en las tecnologías de la información han provocado un aumento sin precedentes del volumen y el flujo de datos. Códigos de barra, tarjetas electrónicas, sensores remotos, transacciones bancarias, satélites espaciales o el reciente proyecto de cooperación internacional para descifrar el código genético humano son ejemplos de la necesidad de filtrar, analizar e interpretar volúmenes de datos que se miden en terabytes.

Consolidación.

Marketing dirigido, modelos de financiación, bonos del estado, control de calidad, la industria del petróleo, banca, biología molecular, prevención de incendios forestales, criminología, diagnósticos médicos, etc. El amplio campo de actuación del que ya disfrutan las herramientas de la Minería de Datos frente a su relativa novedad hace ver áreas problemáticas rápidamente. Los factores principales en la consolidación de las técnicas de la minería de datos como herramienta principal en la toma de decisiones de negocio son:

1. **El sorprendente desarrollo tecnológico unido a los costes de producción y comercialización**. Los avances alcanzados en el procesamiento paralelo (sin duda factor clave) junto con la capacidad de los dispositivos de almacenamiento, permiten hoy día la aplicación a grandes bases de datos de métodos computacionalmente muy complejos en poco tiempo, algo impensable hace 20 años. En [3] se describe un ejemplo muy significativo: el *Banco de América* gastaba 2430 dólares por una consulta sobre 15 gigabytes en el año 1985; en 1995 gastaba 24 dólares por una consulta de 800 gigabytes. Según [4] el almacenamiento de un terabyte de información costaría hace 5 años 10 millones de dólares, hoy en día no llega al millón de dólares.

2. **El aumento en la frecuencia y cantidad en adquisición de datos**. Se estima que actualmente un satélite produce 50 gigabytes por hora [5]. Este ejemplo señala el enorme crecimiento en el trafico de datos, causa no solo del abaratamiento de los sistemas de almacenamiento masivo, sino también de la automatización de muchos experimentos y técnicas de recogida de datos.

3. Intensa y creciente competencia en un comercio saturado. El comercio electrónico es ya una realidad en Estados Unidos. Grandes compañías han visto crecer y aumentar sus puntos de venta como nunca, pudiendo llegar a mercados antes no previstos. Las campañas publicitarias a través del correo electrónico garantizan que el cliente recibe la información y reduce costes implícitos en la publicidad tradicional.

4. **Consolidación de la tecnología para el desarrollo de sistemas de gestión de la información.**

Definición.

Dos términos empleados erróneamente sinónimos son KDD y ***Data Mining***. Según [1] se denomina **Minería de Datos** al conjunto de técnicas y herramientas aplicadas al proceso no trivial de extraer y presentar conocimiento implícito, previamente desconocido, potencialmente útil y humanamente comprensible, a partir de grandes conjuntos de datos, con objeto de:

Predecir de forma automatizada tendencias y comportamientos.
Describir de forma automatizada modelos previamente desconocidos.

KDD (***Knowledge Discovery in Databases***, ***Descubrimiento de Conocimiento en Bases de Datos***) debe referirse al amplio proceso de búsqueda de un alto nivel de conocimiento a partir de datos a un bajo nivel, con la aplicación de los métodos específicos de **Minería de Datos** como uno de los pasos más importantes en todo el proceso global.

KDD supone la convergencia de distintas disciplinas de investigación, fundamentalmente: Aprendizaje Automático, Estadística, Reconocimiento de Patrones, Inteligencia Artificial, Sistemas de Gestión de Bases de Datos, Técnicas de Visualización de Datos, los Sistemas para el Apoyo a la Toma de Decisiones, la Recuperación de Información.

Así, mientras que el término ***Data Mining*** suele utilizarse por estadísticos, analistas de datos y la comunidad de sistemas de gestión de la información (MIS), **KDD** (formalizado en un encuentro celebrado en 1989 [1]) es aplicado a las disciplinas de la **Inteligencia Artificial y el Aprendizaje Automático** que tratan la extracción de conocimiento en grandes bases de datos, donde:

> Dado un conjunto de hechos (datos) H, un lenguaje L y alguna medida de la certeza C, definimos una regularidad o patrón S (pattern) como una sentencia en L que describe relaciones dentro de un subconjunto H_s de H con una certidumbre c, de forma que **S es más sencillo que la enumeración de todos los hechos de H_s.** Una regularidad que sea interesante y altamente cierta (según criterios definidos por el usuario experto) será considerado conocimiento, de forma que cuando el conocimiento se extrae partiendo de los datos de una base de datos, se tiene ***KDD*** [5].

Los pasos en el proceso global del **KDD** no están claramente diferenciados y lo convierten en un proceso iterativo e interactivo con el usuario experto. Las interacciones entre las decisiones tomadas en diferentes pasos, así como los parámetros de los métodos utilizados y la forma de representar el problema suelen ser extremadamente complejos. Pequeños cambios en una parte pueden afectar fuertemente al resultado final. En [5] se estructura el proceso global en 9 etapas:

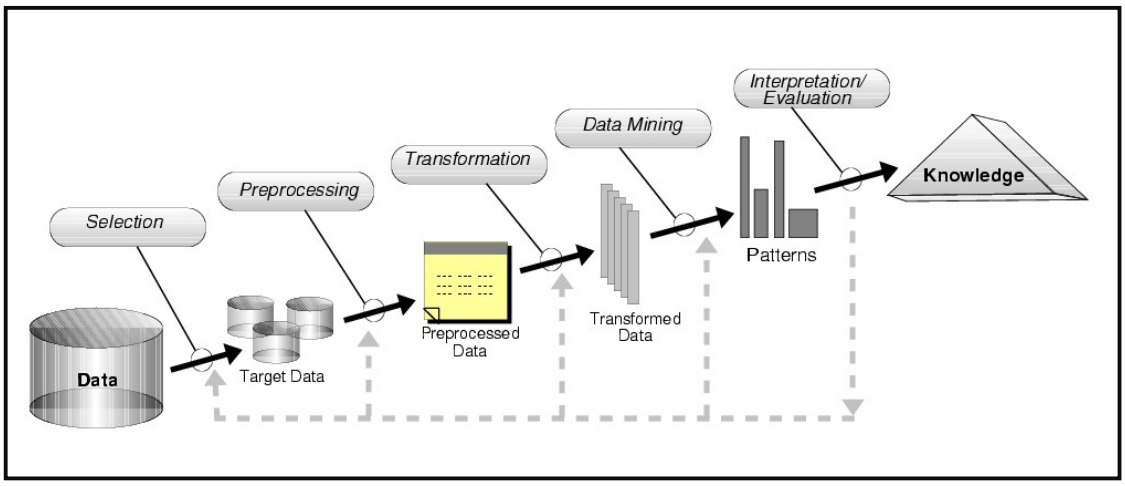


Figura 2. Proceso global del **KDD**.

1. Desarrollo y entendimiento del dominio de la aplicación, el conocimiento relevante y los objetivos del usuario final. Este paso requiere cierta dependencia usuario/analista, pues intervienen factores como:

   Conocer los cuellos de botella del dominio.
   Saber qué partes son susceptibles de un procesado automático
   Cuáles son los objetivos y los criterios de rendimiento exigibles.
   Para qué se usarán los resultados que se obtengan.
   Compromisos entre simplicidad y precisión del conocimiento extraído, etc.

2. Creación del conjunto de datos objetivo, seleccionando el subconjunto de variables o ejemplos sobre los que se realizará el descubrimiento. Esto implica consideraciones sobre:

   La homogeneidad de los datos
   Variación a lo largo del tiempo de los datos
   Estrategia de muestreo.
   Grados de libertad, etc.

3. Limpieza y Preprocesado de los datos: eliminación de ruido, estrategias para manejar valores ausentes, normalización de los datos, etc.

4. Transformación y reducción de los datos. Incluye la búsqueda de características útiles de los datos según sea el objetivo final, la reducción del número de variables y la proyección de los datos sobre espacios de búsqueda en los que sea más fácil encontrar una solución. Este es un paso crítico dentro del proceso global, que requiere un buen conocimiento del problema y una buena intuición, y que, con frecuencia, marca la diferencia entre el éxito o fracaso de la Minería de Datos.

5. Elección del tipo de sistema para Minería de Datos según el conocimiento objetivo que se quiera extraer en el proceso de ***KDD***: clasificación de conceptos, agrupamiento de conceptos, predicción de comportamientos, detección de desviaciones, etc.

6. Elección del algoritmo de **Minería de Datos** según el conocimiento objetivo que se desea extraer.

7. Aplicación del algoritmo de **Minería de Datos** elegido. En este paso se realiza la búsqueda de conocimiento con una determinada representación del mismo. El éxito de la **Minería de Datos** depende en gran parte de la correcta realización de los pasos previos, por parte del usuario.

8. Interpretación del conocimiento extraído, con posibilidad de iterar de nuevo desde el primer paso. La obtención de resultados aceptables dependerá de factores como: definición de medidas del interés del conocimiento (de tipo estadístico, en función de su sencillez, etc.) que permitan filtrarlo de forma automática, existencia de técnicas de visualización para facilitar la valoración de los resultados o búsqueda manual de conocimiento útil entre los resultados obtenidos.

9. Consolidación del conocimiento descubierto, incorporándolo al sistema, o simplemente documentándolo y enviándolo a la parte interesada. Esta etapa incluye la revisión y resolución de posibles inconsistencias con otro conocimiento extraído previamente.

Por todo ello y sin quitar importancia a ninguna etapa del KDD, se puede decir que la Minería de los Datos es la parte fundamental y en la que más esfuerzos se han realizado.

Medidas del Rendimiento.

La definición anterior utiliza términos que son claves para determinar el rendimiento de un sistema de adquisición de conocimiento:

Precisión.

Los descubrimientos representan el contenido de la base de datos que, como reflejo de la realidad, puede contener **imperfecciones y ruido**. Por tanto, será raro que algún conocimiento se cumpla con todos los datos. El grado de certidumbre medirá el **crédito o confianza** que el sistema o usuario puede asignar a cierto descubrimiento; si la **certeza** no es lo suficientemente alta, los patrones descubiertos no llegarán a ser conocimiento.

Interés.

Aunque es posible extraer numerosos **patrones** de cualquier base de datos, sólo se consideran como conocimiento aquéllos que resulten interesantes según ciertos criterios del usuario. En particular, un patrón interesante debe ser nuevo, potencialmente útil y no trivial.

Lenguaje de alto nivel.

El conocimiento descubierto debe representarse de forma inteligible desde el punto de vista humano

Limitaciones.

La inducción de conocimiento en bases de datos se encuentra con dificultades no previstas por los tradicionales sistemas de aprendizaje, pues en el mundo real, las bases de datos suelen ser **dinámicas, incompletas, ruidosas y muy grandes** [1]. Por estos motivos, muchos algoritmos de aprendizaje no son **eficientes** al ser aplicados sobre determinadas bases de datos y gran parte del trabajo que se realiza en la inducción de conocimiento trata de solucionar estos problemas:

Datos dinámicos.

En la mayoría de las bases de datos, los datos son modificados de forma continua. Cuando el valor de los datos almacenados es función del tiempo, el conocimiento inducido varía según el instante en que se obtenga. Por ello es deseable un sistema que funcione de forma continua, en modo secuencial, para tener siempre actualizado el conocimiento extraído.

Datos incompletos.

El manejo de datos incompletos en una base de datos puede deberse a pérdida de valores de algún atributo (al que se asigna entonces el valor desconocido), o a la ausencia del mismo en la vista que el sistema posee sobre los datos. En ambos casos, la incidencia en el resultado dependerá de que el dato incompleto sea relevante o no para el objetivo del sistema de aprendizaje. Por ejemplo, un sistema para aprender a diagnosticar arritmias cardíacas no se verá afectado por la pérdida de datos como el color del pelo del paciente, pero sí por otros como el ritmo cardíaco.

Ruido e incertidumbre en los datos.

El ruido existente en los datos viene determinado tanto por el tipo de valores de los atributos: real (presión arterial), entero (ritmo cardíaco), cadena de caracteres (nombre del paciente) o nominal (tipo de arritmia), como por la exactitud en la medida de dichos valores (especialmente para atributos numéricos). Por otro lado, es frecuente que las regularidades que se encuentran entre los datos de partida no se verifiquen siempre, sino con cierta probabilidad. Este efecto se debe no sólo a la presencia de ruido en la base de datos, sino también a la indeterminación existente en muchos aspectos de la realidad y a imprecisiones en el modelado de la misma (no hay que olvidar que en el diseño de la base de datos se modela la realidad, y todo modelo no es sino una aproximación más o menos precisa).

Eficiencia.

Un algoritmo se considera eficiente cuando su tiempo de ejecución y el espacio de memoria requerido crecen de forma polinómica con el tamaño de los datos de entrada. No es posible aprender de forma eficiente cualquier concepto booleano (problema NP-completo), sin embargo, sí existen algoritmos eficientes para clases restringidas de conceptos, como los representables en forma conjuntiva, etc. Otra posibilidad es el uso de heurísticos y algoritmos aproximados para la inducción de conocimiento.

Tamaño de las bases de datos.

El tamaño de las bases de datos suele ser muy superior al de los conjuntos de entrenamiento de muchos sistemas de aprendizaje, por ello, en las bases de datos muy grandes es inabordable un análisis completo de todos los datos, y deben emplearse técnicas específicas que aceleren el aprendizaje sobre las mismas. Esto marca una importante diferencia entre los sistemas de inducción aplicables en uno y otro caso. Así para aumentar la eficiencia del sistema:

Elegir el algoritmo de aprendizaje con menor complejidad algorítmica, para que los requerimientos de tiempos de computación no crezcan más que de forma polinómica (con un pequeño) grado al aumentar el tamaño de la base de datos.

Utilizar una heurística que oriente la búsqueda de nuevo conocimiento, evitando realizar búsquedas exhaustivas.

Realizar algún tipo de muestreo dentro de la base de datos, para trabajar con una muestra del total. De este modo puede abordarse el análisis de bases de datos muy grandes, a costa de introducir incertidumbre en los resultados de la inducción.

Sistemas de Aprendizaje: Forma, Esfuerzo y Objetivo.

Desprestigiada en sus orígenes por la estadística al moldear suficientemente los datos hasta que los mismos confirmasen lo que se quería postular, la **Minería de Datos** es un proceso que invierte la dinámica del método científico:

> En el método científico, primero se formula la hipótesis y luego se diseña el experimento para coleccionar los datos que confirmen o refuten la hipótesis. Si esto se hace con la formalidad adecuada (cuidando cuáles son las variables controladas y cuáles son las experimentales), se obtiene un nuevo conocimiento.
>
> En la minería de datos, se coleccionan los datos y se espera que de ellos emerjan hipótesis. Se quiere que los datos describan o indiquen por qué son y como son. La más inocente mirada a los datos por un humano, puede inspirarle una hipótesis. Recuérdese que los humanos tenemos grandes poderes de generalización e identificación de patrones. Luego entonces, validar esa hipótesis inspirada por los datos en los datos mismos, será numéricamente significativa, pero experimentalmente inválida. De ahí que la minería de datos deba presentar un enfoque exploratorio, y no confirmador. Usar la minería de datos para confirmar nuestras hipótesis puede ser peligroso, pues estamos haciendo una inferencia poco válida.
>
> Afortunadamente, las técnicas de validación desarrolladas en la década de los 80 en el Aprendizaje Automático, hacen posible que las inferencias de la Minería de Datos se validen para obtener patrones realmente ciertos y no sólo reflejos de una manipulación de los datos.

Clasificación.

Dos clasificaciones clásicas de los sistemas de aprendizaje distinguen el modo y el esfuerzo realizado en la adquisición de conocimiento [6].

Modo de aprendizaje.

El primer grupo distingue dos claras tendencias dadas por la psicología: el enfoque conductista y el enfoque cognoscitivo. Esto marca diferentes actitudes de los sistemas ante el proceso de aprendizaje, así como su empleo en diferentes aplicaciones y el uso de diferentes lenguajes para representar el conocimiento.

*Sistemas Conductistas***.** Según la psicología conductista, el aprendizaje es la capacidad de experimentar cambios adaptativos para mejorar el rendimiento, por lo que un sistema de aprendizaje será como una *caja negra* capaz de adecuar su comportamiento para que el rendimiento de sus respuestas ante los datos de entrada aumente durante el proceso de aprendizaje. Los sistemas de aprendizaje conductistas hacen mayor énfasis en modelos de comportamiento que en la representación interna del conocimiento.

Los lenguajes de descripción suelen ser diferentes para los objetos y para el conocimiento: los objetos se describen por vectores de características, mientras que para el conocimiento se emplean parámetros o tablas. Esta representación del conocimiento hace difícil su traducción a reglas que expliquen de forma racional el comportamiento del sistema. Entre los sistemas conductistas de aprendizaje, hay que destacar los sistemas conexionistas y los sistemas evolucionistas, que realizan inducción de conocimiento.

Sistemas cognoscitivos. Según el enfoque cognoscitivo, el aprendizaje consiste en la construcción y modificación de la representación del conocimiento. Durante el proceso de aprendizaje se produce un incremento del conocimiento, que supone un cambio cuantitativo y cualitativo. La **calidad** del aprendizaje vendrá dada no sólo por el aumento de precisión del conocimiento almacenado (en una base de conocimiento), sino también por la utilidad del mismo para los objetivos del usuario y por el nivel de abstracción empleado. Por tanto, la representación del conocimiento jugará un papel principal en los sistemas que sigan este enfoque.

Los lenguajes de descripción usados por estos sistemas suelen coincidir para representar a los objetos y al conocimiento. Están basados, normalmente, en la lógica (de proposiciones o de predicados) o en representaciones estructuradas (como los marcos). Las aplicaciones de los sistemas de aprendizaje cognoscitivo dependen del tipo de aprendizaje que realicen, siendo los más importantes los que utilizan deducción (sistemas EBL, basados en explicaciones), analogía (sistemas expertos basados en casos) o inducción (adquisición y formación de conceptos).

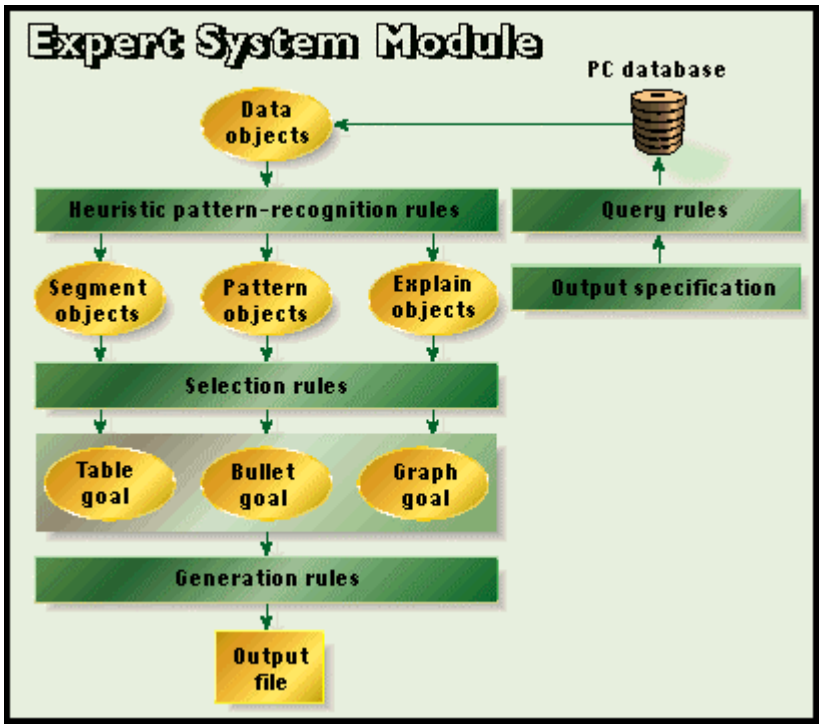


Figura 3. Sistema Experto.

Este documento repasa brevemente las principales disciplinas relacionadas con el aprendizaje inductivo.

Esfuerzo en el aprendizaje.

En cualquier proceso de aprendizaje, el aprendiz aplica el conocimiento poseído a la información que le llega, para obtener nuevo conocimiento que es almacenado para poder ser usado posteriormente. El esfuerzo requerido por el aprendiz se traduce al número de inferencias que necesita sobre la información que tiene disponible. Bajo este punto de vista han sido identificadas varias estrategias, aunque en la práctica, muchos procesos de aprendizaje aplican de forma simultánea varias de ellas. Según [7] una clasificación en orden creciente de esfuerzo inferencial por parte del aprendiz, es la siguiente:

1. Aprendizaje por implantación directa (***Rote Learning***): Es un caso extremo, en el que el aprendiz no ha de realizar ningún tipo de inferencia sobre la información suministrada, sino que la acepta directamente. Esta estrategia incluye aprendizaje por programación y por memorización.

2. Aprendizaje por instrucción: El sistema de aprendizaje adquiere el nuevo conocimiento a través de la información proporcionada por un maestro, pero no la copia directamente en memoria, sino que selecciona los datos más relevantes y/o los transforma a una forma de representación más apropiada.

3. Aprendizaje por deducción: Partiendo del conocimiento suministrado y/o poseído, se deduce el nuevo conocimiento, es decir, se transforma el conocimiento existente mediante una función preservadora de la verdad.

4. Aprendizaje por analogía: Se adquiere un nuevo concepto mediante la modificación de la definición ya conocida de un concepto similar. El aprendizaje por analogía puede ser entendido como una combinación de la inducción y la deducción, ya que mediante la inferencia inductiva se determinan características comunes a los dos conceptos comparados, unificando la misma definición para ambos; entonces se aplica la deducción para obtener las características esperadas para el nuevo concepto. Este tipo de aprendizaje es especialmente importante en la resolución de problemas.

5. **Aprendizaje por inducción**: El sistema de aprendizaje aplica la inducción a los hechos u observaciones suministradas, para obtener nuevo conocimiento. La inferencia inductiva no preserva la verdad del conocimiento, sólo su falsedad; es decir, si partimos de hechos falsos, el conocimiento adquirido por inducción será falso, pero si los hechos son verdaderos, el conocimiento inducido será válido con cierta probabilidad (y no con certeza absoluta, como ocurre con la deducción). Hay dos tipos de aprendizaje inductivo:

   - **Aprendizaje con ejemplos**: el nuevo conocimiento es inducido mediante la generalización a partir de una serie de ejemplos y contraejemplos. Este método también se conoce como **adquisición de conceptos o aprendizaje supervisado**.

   - **Aprendizaje por observación y descubrimiento**: el sistema de aprendizaje analiza una serie de entidades y determina características comunes, que pueden ser agrupadas formando un concepto previamente desconocido. Se conoce como **formación de conceptos o aprendizaje no supervisado**.

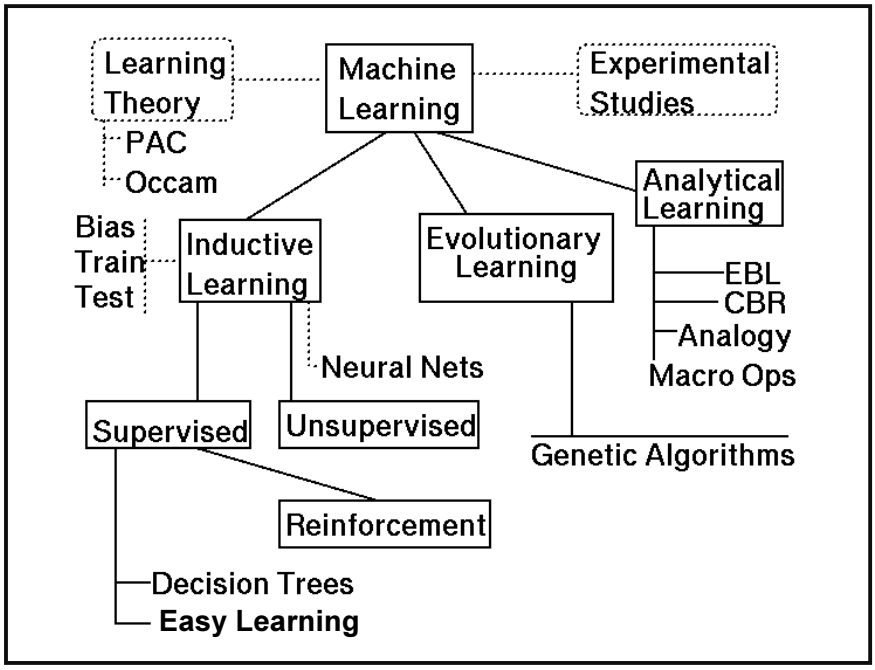


Figura 5. Clasificación de los sistemas de aprendizaje.

De las distintas disciplinas que convergen en el KDD es el aprendizaje automático (Machine Learning) la que da el soporte principal para la construcción de las herramientas de la Minería de Datos. La figura 3 muestra un diagrama donde los dos criterios anteriores (esfuerzo y forma en el aprendizaje) estructuran el conjunto de todas las disciplinas en una única clasificación. El carácter interdisciplinario queda claramente reflejado.

Estructura.

Según [5] son tres los componentes principales en un algoritmo de ***Data Mining***:

El Modelo de Representación: el lenguaje utilizado para representar el conocimiento (patrones descubiertos). Si la representación es muy limitada la precisión obtenida será siempre reducida independientemente del tiempo o el número de ejemplos para el entrenamiento. Se distinguen:

- Representaciones basadas en la lógica (de proposiciones o de predicados).
- Representaciones no simbólicas.
- Representaciones estructuradas.
- Representaciones basadas en ejemplos.

El Modelo de Evaluación: funciones de ajuste que determinan de forma cuantitativa como de bueno es un patrón descubierto respecto los objetivos iniciales del proceso de **KDD**.

El Método de Búsqueda: consistente en el modelo de búsqueda y los parámetros de búsqueda. Fijados los modelos de representación (o la familia de representaciones) y de evaluación el problema queda reducido encontrar el modelo y los parámetros de búsqueda optimizar el criterio de evaluación.

Objetivos.

Clasificación y Regresión. Son los sistemas de adquisición de conocimiento más extendidos. Dado un conjunto de N ejemplos (registros) con M atributos (campos) y A etiquetas (clases de ejemplos) se obtiene una función que describe las características de los datos para cada etiqueta, mapeando nuevos datos cuyas clases se desconocen. La Regresión determina relaciones existentes entre las variables (atributos) y obtiene una función de predicción.

Agrupamiento de conceptos. El ***Clustering*** realiza la búsqueda de un conjunto finito de categorías que describen a los datos de entrada y en las que se agrupan. La segmentación suele realizarse estadísticamente o utilizando redes neuronales. Es muy utilizado en marketing para encontrar grupos afines de clientes, así como en medicina donde el objetivo es encontrar pacientes con perfiles comunes. Suelen apoyarse en un tipo concreto de redes neuronales llamadas redes *Kohonen* o algoritmos demográficos.

Compactación: búsqueda de descripciones más compactas de los datos.

Modelado de dependencias: se obtiene un modelo que describe las dependencias significativas entre las variables de los datos.

Detección de desviaciones: búsqueda de desviaciones importantes de los datos respecto de valores anteriores o medios, etc.

Aprendizaje Supervisado.

Para describir los distintos algoritmos usaremos un clásico ejemplo didáctico (tabla 1) donde se describen las condiciones ambientales que han determinaron en el pasado si se jugó o no un partido de golf. Las 4 primeras columnas representan los atributos de la base de datos; los valores pueden ser de distinto tipo (discreto o continuo) y estar en distinto rango. La ultima columna es la etiqueta (o clase) que diferencia a cada uno de los ejemplos (o casos) y sobre la cual se intenta extraer conocimiento.

outlook	temp	humid	wind	Class
sunny	75	70	true	play
sunny	80	90	true	no play
sunny	85	85	false	no play
sunny	72	95	false	no play
sunny	69	70	false	play
overcast	72	90	true	play
overcast	83	78	false	play
overcast	64	65	true	play
overcast	81	75	false	play
rainy	71	80	true	no play
rainy	65	70	true	no play
rainy	75	80	false	play
rainy	68	80	false	play
rainy	70	96	false	play
overcast	76	88	true	?

Tabla 1. Base de datos de ejemplo.

El último ejemplo representa la situación ambiental actual para la que se quiere conocer si en función de los datos pasados se jugará o no el partido.

Arboles de decisión.

Es una forma de representación sencilla basada en la lógica de proposiciones extendida (también denominada "0+" o *objeto-atributo-valor*). Utiliza la técnica *"divide y vencerás"* para clasificar el conjunto de ejemplos en un número finito de clases predefinidas realizando particiones o subconjuntos de forma recursiva y sin backtracking [8].

Las particiones del conjunto inicial se traducen en subarboles para cuyos ejemplos, los valores del nodo padre cumplen la condición o etiqueta de la rama asociada, siendo conjuntos disjuntos respecto al atributo sobre el cual se generan los cortes.

Una vez se obtiene el árbol a partir del conjunto de entrenamiento, los ejemplos del test cuya clase se desconoce son clasificados recorriendo el árbol por aquellas ramas para las cuales los valores de sus atributos se cumplen hasta alcanzar una hoja. Los nodos del árbol quedan etiquetados con nombres de atributos, las ramas con los valores posibles para el atributo, y las hojas con las diferentes clases.

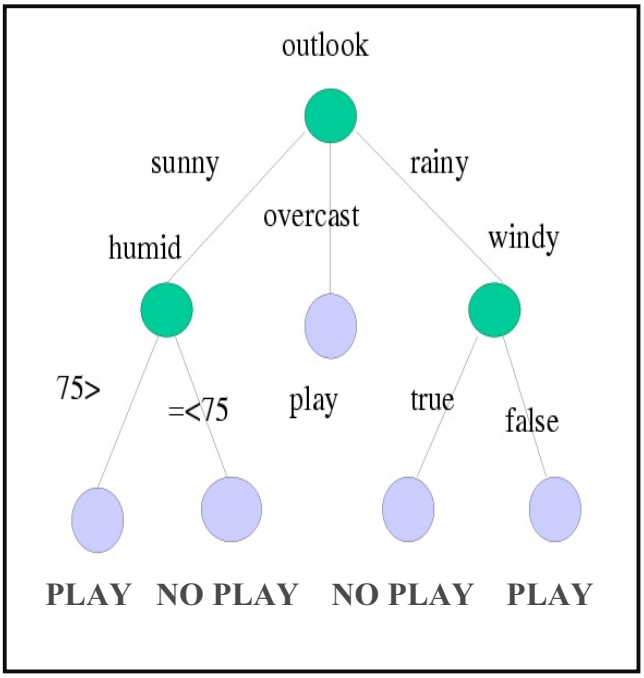


Figura 6. Árbol de decisión.

Los sistemas basados en árboles de decisión forman una familia llamada TDIDT (Top-Down Induction of Decision Trees), cuyo representante más conocido es C4.5 [9], mejora de ID3 (Interactive Dichotomizer) que permite clasificar ejemplos con atributos que toman valores continuos. El método de clasificación de C4.5 es ya una técnica empleadas por herramientas de ***Data Mining*** junto a otros dos métodos específicos: Arboles de Clasificación y Regresión (CART: Classification And Regression Tree) y Detección de Interacción Automática de Chi Cuadrado (CHAI: Chi Square Automatic Interaction Detection).

La principal ventaja de esta representación son sus reducidos requisitos computacionales. Sin embargo cuando el número de atributos es elevado y para estos lo es también el dominio de valores, se vuelven difíciles de comprender. La construcción y clasificación de un árbol de decisión viene dada por: la función de selección, el criterio de parada y el algoritmo de podado.

La función selección es el criterio utilizado por el árbol para particionar el conjunto de entrenamiento, esto es, obtener los sucesivos subarboles que lo componen. Las mas utilizada es la ganancia de información media (**Gain ratio** [10]) que selecciona para cada partición el atributo que genera la reducción máxima de la entropía media. Otras medidas utilizadas son GINI [11 – Cart 86] y la ganancia de información normalizada [12].

El criterio de parada determina cuando una partición se completa. Generalmente se llega a una hoja cuando todos los ejemplos del subconjunto que contiene pertenencen a la misma clase, pero no es el único criterio.

El algoritmo de podado intenta prevenir el sobreajuste (overfiting). El sobreajuste puede evitarse mediante un pre-podado (que sufre el efecto horizonte) o un post-podado donde el árbol ya esta totalmente construido (Error-Complexity [11], error pesimista [8], error mínimo).

En su contra los arboles de decisión poseen una potencia descriptiva limitada por las condiciones o reglas con las que se divide el conjunto de entrenamiento (normalmente relaciones de comparación entre un atributo y un valor). Los *arboles monothetic* (un solo atributo para cada nodo) están limitados a particiones *axis-parallel* del espacio de búsqueda. Los *arboles polythetic* (más de un atributo para cada nodo) tienen un alto coste computacional.

```
TDIT(Ejemplos) ← fPodado(fGenerarArbol(Ejemplos));

fGenerarArbol(Ejemplos)
        Si  fCondicionParada(Ejemplos)
                → ClaseMayoritaria(Ejemplos);
        Sino
                mejorCorte ← fSeleccion(Ejemplos);
                Nodo ← fNuevoNodo(mejorCorte);
                ∀ valor v ∈ Ejemplos(mejorCorte)
                        Corte_V ← { e ∈ Ejemplos / e(mejorCorte) = v }
                        SubArbol_V ← fGenerarArbol(Corte_V)
                        Nodo.arbol_V ← SubArbol_V
```

Figura 7. TDIT.

Reglas de producción.

Una desventaja de los árboles de decisión es que tienden a ser demasiado grandes en aplicaciones reales (al manejar valores continuos y decenas de atributos) y por tanto, se hacen difíciles de interpretar desde el punto del usuario. Para simplificar y hacer comprensible un modelo tan complejo existen técnicas para convertir árboles de decisión en otras formas de representación, como las reglas de producción del tipo *si-entonces*, basadas también en lógica de proposiciones: el consecuente es una clase, y el antecedente es una expresión proposicional, que puede estar en forma normal conjuntiva (CNF) o ser simplemente un término (figura 3). Sin embargo sigue siendo una forma de representación limitada ya que no permite expresar incertidumbre.

Play ← outlook = sunny, humid > 75
Play ← outlook = rainy, wind = FALSO
Play ← outlook = overcast

No Play ← outlook = sunny, humid ≤ 75
No Play ← outlook=rainy, wind = CIERTO

Figura 8. Reglas de producción.

En [9] se propone una técnica para construir reglas de decisión, basadas en lógica de proposiciones, a partir de árboles de decisión, convirtiendo cada ruta *raiz-hoja* en una regla y eliminado aquellas redundantes. El problema es que incluso las reglas de producción que se obtienen resultan a veces demasiado complejas para el experto.

Algunos sistemas como PRISM [14], generan directamente reglas algo más sencillas, sin tener que construir el árbol previamente, mediante un algoritmo parecido a ID3 y con una precisión similar.

La familia AQ [7] la forman sistemas (AQ11, AQ15, etc.) que generan descripciones estructurales, por diferenciarlas de las descripciones de atributos de los sistemas anteriormente mencionados. Estas descripciones son también reglas de producción, aunque con mayor capacidad descriptiva, pues su antecedente es una fórmula lógica. La notación utilizada en estas reglas se denomina VL1 (Variable-valued Logic system 1, [15]) y permite utilizar selectores (igualdad entre un atributo y un valor o conjunto de valores), complejos (conjunciones de selectores) y disyunciones de complejos para construir las reglas de producción.

Listas de decisión.

Las listas de decisión son otra forma de representación basada en lógica de proposiciones. Es una generalización de los árboles de decisión y de las representaciones conjuntivas (CNF) y disyuntivas (DNF). Una lista de decisión es una lista de pares de la forma:

(d_1, C_1);
(d_2, C_2);
…
(d_n, C_n);

donde cada d_i es una descripción elemental, cada C_i es una clase, y la última descripción d_n es el valor verdadero. Así, la clase de un objeto será C_i cuando d_i sea la primera descripción que lo satisface. Por tanto, se puede pensar en una lista de decisión como en una regla de la forma:

si d_1 entonces C_1, sino
si d_2 entonces C_2, sino
…
si d_n entonces C_n.

Se usan por ejemplo en el sistema CN2 [16] que es una modificación del sistema AQ, que pretende mejorar el aprendizaje a partir de ejemplos con ruido (al evitar la dependencia de ejemplos específicos e incorporar una poda del espacio de búsqueda). Las descripciones elementales de los pares que forman la lista de decisión tienen la misma forma que los complejos de AQ.

(outlook = rainy AND wind, No Play);
(outlook = sunny AND humid ≤ 75, No Play);

(CIERTO, Play)

Figura 9. Listas de decisión.

Representaciones basadas en ejemplos.

El aprendizaje basado en ejemplos (***Instance-Based Learning Algorithms, IBL Algorithms***) ha recibido distintos nombres desde sus origines: *Lazy-Learning, Instance-Based, Memory_based, Exemplar-Based,* etc. Representan el conocimiento mediante ejemplos representativos, basándose en "similitudes" entre los datos [17].

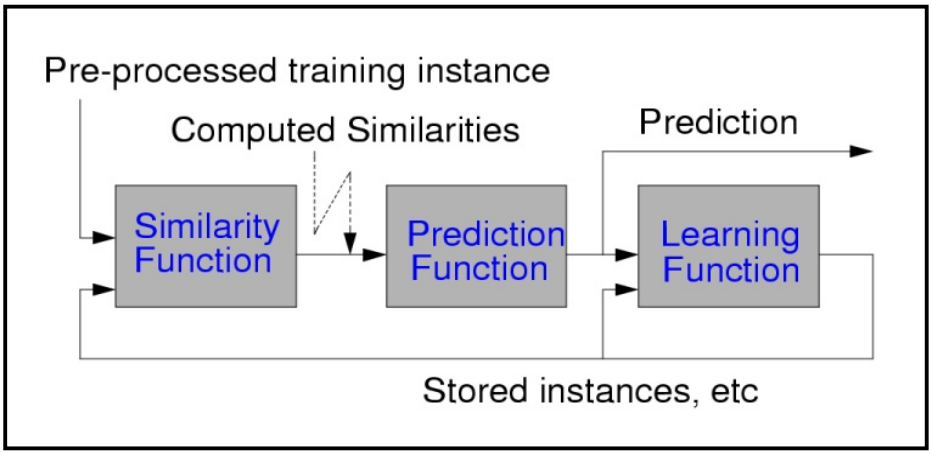


Figura 10. Lazy Learning.

El aprendizaje consiste en la selección de los ejemplos que mejor representan a los conceptos existentes en la base de datos; estos ejemplos representativos serán los únicos que se almacenen, reduciendo así considerablemente el espacio necesario y acelerando el posterior proceso de aprendizaje.

Los sistemas de aprendizaje por vecindad suelen realizar un preprocesado de los datos antes del aprendizaje y un postprocesado. Normalmente el preprocesado consiste en filtrar el ruido y normalizar (lineal o standard) en un mismo rango todos los atributos del conjunto de entrada. Sistemas más complejos utilizan una representación diferente (prototipos, hiperrectangulos, reglas, etc.) para reducir de entrada el espacio de búsqueda.

Utilizan una **Función Similitud** para clasificar los nuevos ejemplos según sea su parecido o proximidad. Esta suele ser difícil de definir cuando la base de datos contiene atributos discretos y continuos, utilizando distintas métricas en cada caso. La tabla 2 muestra un resumen de las principales distancias continuas utilizadas. En [18] se propone una métrica homogénea.

Minkowsky: $D(\boldsymbol{x},\boldsymbol{y}) = \left(\sum_{i=1}^{m} |x_i - y_i|^r \right)^{1/r}$

Euclidean: $D(\boldsymbol{x},\boldsymbol{y}) = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2}$

Manhattan / city-block: $D(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} |x_i - y_i|$

Camberra: $D(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} \frac{|x_i - y_i|}{|x_i + y_i|}$

Chebychev: $D(\boldsymbol{x},\boldsymbol{y}) = \max_{i=1}^{m} |x_i - y_i|$

Quadratic: $D(\boldsymbol{x},\boldsymbol{y}) = (\boldsymbol{x}-\boldsymbol{y})^T Q(\boldsymbol{x}-\boldsymbol{y}) = \sum_{j=1}^{m} \left(\sum_{i=1}^{m} (x_i - y_i) q_{ji} \right) (x_j - y_j)$

Q is a problem-specific positive definite $m \times m$ weight matrix

Mahalanobis:

$D(\boldsymbol{x},\boldsymbol{y}) = [\det V]^{1/m} (\boldsymbol{x}-\boldsymbol{y})^T V^{-1} (\boldsymbol{x}-\boldsymbol{y})$

V is the covariance matrix of $A_1..A_m$, and A_j is the vector of values for attribute j occuring in the training set instances $1..n$.

Correlation:

$$D(\boldsymbol{x},\boldsymbol{y}) = \frac{\sum_{i=1}^{m} (x_i - \overline{x_i})(y_i - \overline{y_i})}{\sqrt{\sum_{i=1}^{m} (x_i - \overline{x_i})^2 \sum_{i=1}^{m} (y_i - \overline{y_i})^2}}$$

$\overline{x_i} = \overline{y_i}$ and is the average value for attribute i occuring in the training set.

Chi-square: $D(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} \frac{1}{sum_i} \left(\frac{x_i}{size_{\boldsymbol{x}}} - \frac{y_i}{size_{\boldsymbol{y}}} \right)^2$

sum_i is the sum of all values for attribute i occuring in the training set, and $size_x$ is the sum of all values in the vector $\boldsymbol{x}$.

Kendall's Rank Correlation: $D(\boldsymbol{x},\boldsymbol{y}) = 1 - \frac{2}{n(n-1)} \sum_{i=1}^{m} \sum_{j=1}^{i-1} \text{sign}(x_i - x_j)\,\text{sign}(y_i - y_j)$

sign(x)=-1, 0 or 1 if $x < 0$, $x = 0$, or $x > 0$, respectively.

Tabla 2. Distancias continuas usuales.

Los algoritmos de aprendizaje basado en ejemplos han adquirido mayor importancia recientemente con los sistemas de razonamiento basado en casos (***Case-Based Reasoning***) para diagnóstico y resolución de problemas. Además, pueden utilizarse como paso previo a otros sistemas de aprendizaje a partir de ejemplos, para entrenarlos con conjuntos de ejemplos más pequeños y representativos.

Redes neuronales.

Incluidas dentro de los modelos conexionistas, ([19], [20]) son sistemas de representación no simbólico, formados por un conjunto de sencillos elementos de computación llamados neuronas artificiales. Estas neuronas están interconectadas a través de unas conexiones con unos pesos asociados, que representan el conocimiento en la red. Cada neurona calcula la suma de sus entradas, ponderadas por los pesos de las conexiones, le resta un valor umbral y le aplica una función no lineal (normalmente una *sigmoide*).

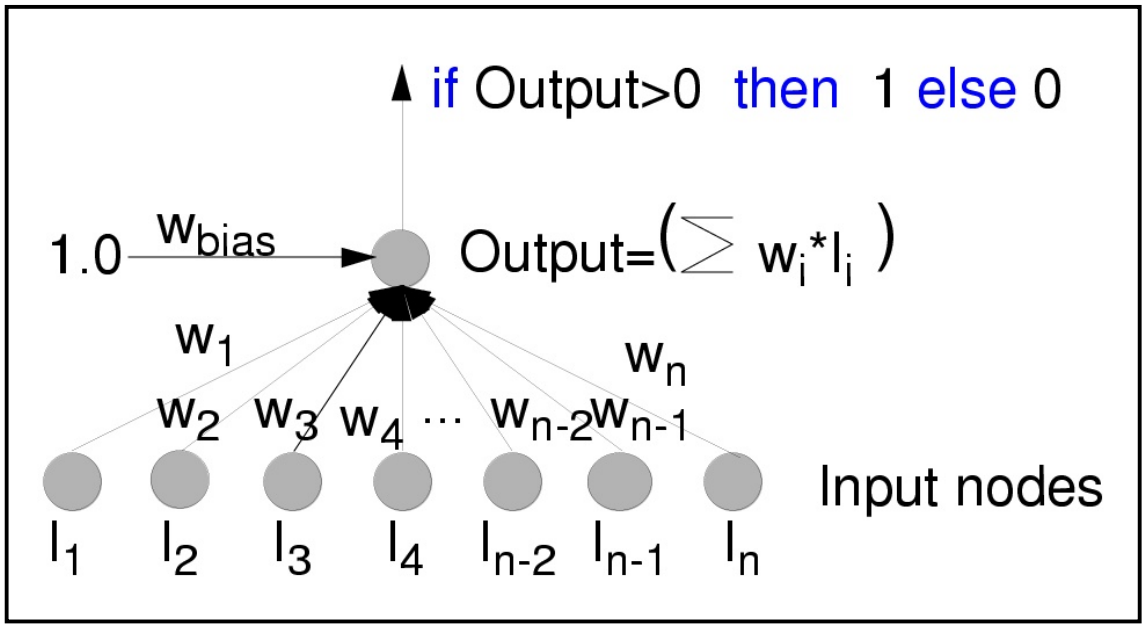


Figura 11. Modelo *Percepton.*

Dos modelos básicos de redes neuronales son el percepton (basado en la teoria de resonancia adaptativa) y el backpropagation (redes recurrentes). En modelos multicapa como el percepton, el resultado sirve de entrada a las neuronas de la capa siguiente. El modelo *back-propagation* utiliza un método iterativo para propagar los términos de error (diferencia entre valores obtenidos y valores deseados), necesarios para modificar los pesos de las conexiones interneuronales. El *back-propagation* puede considerarse como un método de regresión no lineal en el que aplica un descenso de gradiente en el espacio de parámetros (pesos), para encontrar mínimos locales en la función de error [5]. Las redes neuronales han sido utilizadas con éxito en diferentes tipos de problemas:

- Auto-asociación: la red genera una representación interna de los ejemplos aportados, y responde con el más aproximado a su "memoria". Ejemplo: máquina de Boltzman.
- Clasificación de patrones: la red es capaz de clasificar cada entrada en un conjunto predefinido de clases; pe. back-propagation.
- Detección de regularidades: la red se adapta a los ejemplos de entrada, tomando de ellos varias características para clasificarlos; en este caso, el conjunto de clases no está definido de antemano, por lo que el aprendizaje es no supervisado. Ej.: red MAXNET, ART1, mapas de Kohonen, red de Oja, etc.

Las tasas de error de las redes neuronales son equivalentes a las de las reglas generadas por los métodos de aprendizaje simbólicos, aunque son algo más robustas cuando los datos son ruidosos. Las principales desventajas para usar redes neuronales en la **Minería de Datos** son:

- el aprendizaje es bastante más lento que en un sistema de aprendizaje simbólico [21].
- el conocimiento obtenido por las mismas no es representable en forma de reglas inteligibles, sino que lo forma el conjunto de pesos de las conexiones interneuronales.
- además, es difícil incorporar conocimiento de base o interacción del usuario en el proceso de aprendizaje de una red neuronal.
- Requiren gran cantidad de variables (neuronas).

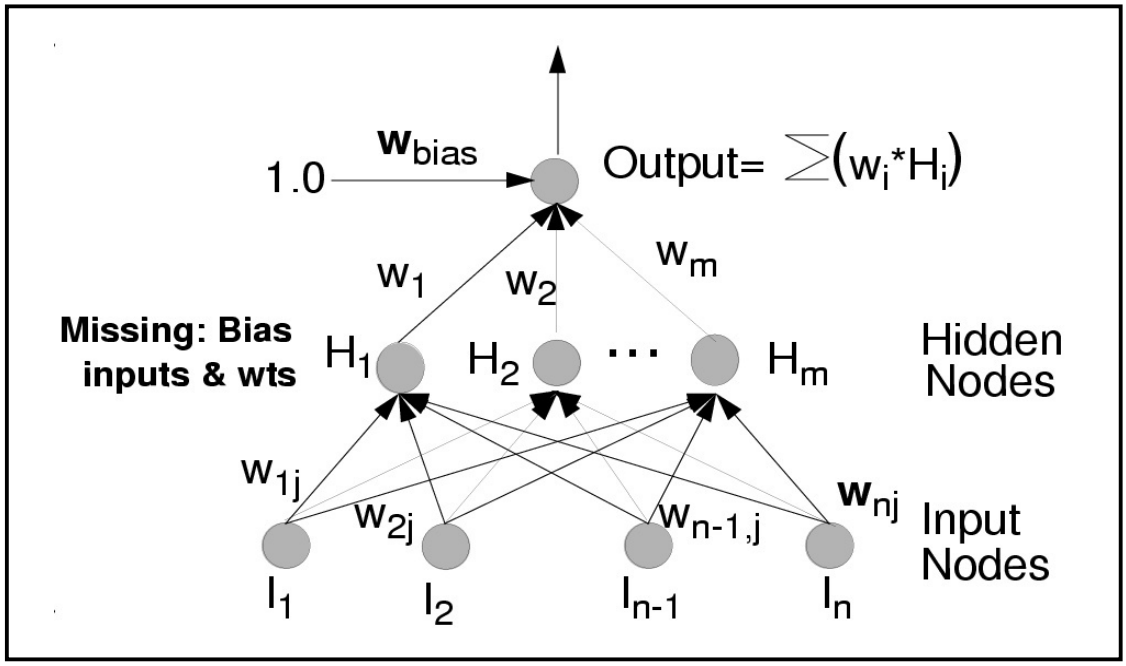


Figura 12. Modelo Backpropagation.

Algoritmos Genéticos.

Los algoritmos genéticos surgen a finales de la década de los 60 gracias a John Holland, entonces investigador de la universidad de Michigan. Inicialmente denominados *planes reproductivos*, se popularizan bajo el nombre de algoritmos genéticos tras la publicación de su libro *Adaptation in Natural and Artificial Systems* en 1975.

Se inspiran en la teoría de la selección de las especies de Charles Darwin que sostiene que aquellos individuos de una población que posean los caracteres más ventajosos dejarán proporcionalmente más descendencia en la siguiente generación; y si tales caracteres se deben a diferencias genéticas que pueden transmitirse a los descendientes, tenderá a cambiar la composición genética de la población, aumentando el número de individuos con dichas características. De esta forma, la población completa de seres vivos se adapta a las circunstancias variables de su entorno. El resultado final es que los seres vivos tienden a perfeccionarse en relación con las circunstancias que los envuelven.

La flexibilidad de los algoritmos genéticos les permiten ser aplicados como técnicas de Búsqueda, Clasificación, Predicción y Optimización (parametrización, configuración, maximización, minimización). Esto es así puesto que una predicción puede considerarse como una clasificación en la que interviene el tiempo, o como una búsqueda dentro del espacio de posibles estados futuros, o como un problema de minimización del tiempo empleado en esta búsqueda.

En este sentido plantean el problema como un proceso iterativo de optimización donde la **función de aptitud (**función objetivo) debe ser capaz de *castigar* a las malas soluciones, y de *premiar* a las buenas, de forma que sean estas últimas las que se propaguen con mayor rapidez.

A partir de un conjunto aleatorio de soluciones tentativas que constituye la población inicial, combinan las mejores respuestas de ese conjunto para crear uno nuevo que reemplaza al anterior, y que se constituye como la generación subsecuente. Repetidamente, con cada nueva iteración, la población se va refinando consiguiendo mejores respuestas al problema, hasta converger finalmente la mayoría de los individuos en un entorno de la solución óptima.

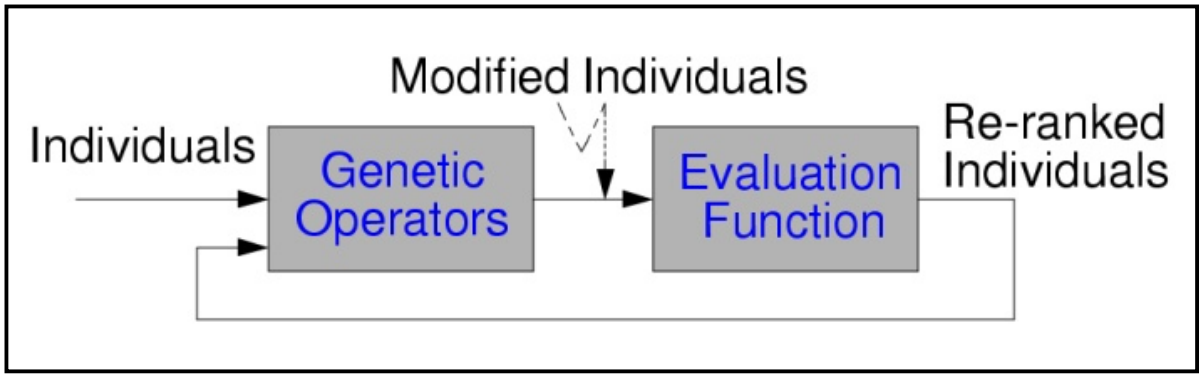


Figura 13. Diagrama para un Algoritmo genético.

Identificado el espacio n-dimensional de posibles soluciones (población inicial) en un lenguaje de representación de conocimiento denominado *Cromosoma* (*Cadena Cromosómica* o *Genotipo*), y a sus diferentes partes *Genes*, aparece la primera generación de la población inicial.

La codificación más común del espacio de soluciones suele realizarse a través de cadenas binarias (propuesto por Holland y de fácil implementación), aunque se han utilizado también números reales y letras.

En el siguiente paso se calcula la función de salud (llamada también función de fuerza) para cada individuo de la población, que indica el grado en que cada solución tentativa es adecuada como solución al problema (en Biología sería el grado de Adaptación del individuo a su entorno). La función fuerza selecciona a los individuos con mayor salud para que generen la siguiente generación. Este grupo constituye los progenitores de la nueva población. Aquellos individuos con muy baja adaptación, es decir, claramente malos como solución al problema, desaparecen al no tener descendencia. Sólo las soluciones más óptimas se reproducen y sus genes pasan de una generación a otra. Combinando las soluciones tentativas del grupo de progenitores se generan nuevas soluciones tentativas, los hijos. La creación de estos nuevos individuos se lleva a cabo mediante 4 operadores genéticos. Los dos principales son:

CRUCE.
Se elige un punto dentro de los cromosomas del padre 1 y del padre 2, y por ahí se cortan sus cromosomas. La cadena cromosómica del hijo se compone de un trozo de la del padre 1 y de la parte complementaria del padre 2. El hijo es una nueva combinación de los genes que ya existen en los padres, y en cierta medida mantiene cierta similitud con sus progenitores, pero es un punto enteramente nuevo en el espacio de soluciones.

MUTACION.
Provoca un cambio aleatorio en algunos de los genes del hijo, introduciendo así nuevos genes en la población, a través de este individuo. Estos cambio serán favorables si aumenta la salud del individuo que los lleva, es decir, si le hacen ser una solución mejor. Si los cambios son perjudiciales, estos genes no se propagarán de una generación a otra, porque los individuos que los portan se extinguirán, al ser peores soluciones nunca serán elegidos como padres y morirán.

```
t = 1;
G ← PoblacionInicial(f_Salud);

Repetir
        t:=t+1;
        S ← Seleccion(G);
        G ← Evolucion(S, fCruce, fMutacion, fSalud);

hasta fParada(G, t);
```

Figura 14. Algoritmo genético.

El proceso se repite hasta que padres e hijos convergen, esto es, la población descendiente no mejora la anterior. Las posibilidades de esta metodología transciende de la búsqueda heurística debido a una razón de peso: el algoritmo no trabaja Top-Down, sino Bottom-Up: no es necesario disponer de un conocimiento profundo del problema a reslover, para que éste pueda ser descompuesto en sub-problemas, sino que se parte de estructuras simples que interactúan, dejando que sea la evolución quien haga el trabajo. Únicamente basta con ser capaces de identificar cualitativamente en que casos los patrones se acercan o se alejan de la solución buscada, para que el sistema se perfeccione automáticamente, desarrollando sus propios comportamientos, funcionalidades y soluciones, esto le permite a los algoritmos genéticos un enorme campo de aplicación:

- Predicción de series temporales
- Parametrización de Sistemas
- Búsqueda de reglas en juegos
- Enrutamiento
- Asígnación de recursos orden de procesos en varias CPU
- Diseño de pistas de circuitos integrados VLSI
- Ordenación y gestión de inventarios (ordenar cajas en un almacén)
- Asígnación de horarios de clase o turnos de trabajo en un hotel
- Resolución de sistemas de ecuaciones no lineales
- Gestión de franquicias
- Toma de decisiones Financieras

COMPARATIVA DE LAS DISTINTAS METODOLOGIAS

La tabla 3 muestra el coste de cada uno de los sistemas de aprendizaje por inducción vistos anteriormente. Los valores se obtuvieron considerando el peor de los casos, por lo que para algunos casos concretos se obtendrían seguramente resultados distintos. La leyenda es la siguiente:

A: Número de atributos.
b: Semánticas diferentes para cada paradigma.
I: Número de iteraciones.
N: Número de ejemplos (Tamaño de la base de datos).
M: Número máximo de nodos para el modelo *backpropagation*
V: Número máximo de valores por atributo

DISCIPLINA	ALGORITMO	ENTRENAMIENTO	TEST	ESPACIO
TDIT	C4.5	$O(A^2N)$	O(LogA)	O(N Logb)
Reglas de Inducción	CN2	$O(A^2Nb^2)$	O(Ab)	O(Ab)
Lazy Learning	1-NN	O(1)	$O(AN^2)$	O(AN)
Algoritmos Genéticos	Genesis	O(IAb)	O(Ab)	O(Ab)
Redes Neuronales	Backpropagation	$O(IMN^2)$	$O(MN^2)$	$O(M^2)$

Tabla 3. Comparativa de la complejidad de los distintos paradigmas de aprendizaje.

Limitaciones.

Aunque las representaciones basadas en lógica de proposiciones han sido usadas con éxito en muchos sistemas de aprendizaje en ordenadores, tienen algunas importantes limitaciones superadas por la lógica de predicados de primer orden [22], que restringen su campo de aplicación:

- **Potencia expresiva**: Las representaciones basadas en lógica de proposiciones limitan la forma de los patrones que pueden ser representados, ya que, en general, no pueden expresar relaciones. Así, por ejemplo, no se pueden representar (al menos de un modo sencillo) patrones en los que se cumpla una relación de igualdad entre dos atributos (sólo se permitiría expresar la igualdad entre un atributo y un valor constante).

- **Conocimiento de base**: Es difícil incorporar conocimiento de base en el proceso de aprendizaje. Una forma sencilla de conocimiento de base la constituyen restricciones impuestas a las descripciones generadas por el sistema, aunque ésto puede resultar demasiado restrictivo.

- **Restricciones en el vocabulario**: Las descripciones de los sistemas actuales vienen limitadas por un vocabulario fijo de atributos proposicionales. Podría ser muy útil tener la posibilidad de mejorar la representación mediante la invención de nuevos predicados.

Estas limitaciones pueden superarse con una representación del conocimiento más potente: la lógica de predicados de primer orden. Cada vez hay más sistemas de aprendizaje en ordenadores que utilizan de algún modo la lógica de primer orden, surgiendo así una nuevo área de interés llamado programación lógica inductiva (***Inductive Logic Programming, ILP***).

El objetivo de la programación lógica inductiva es construir un programa lógico (en lógica de predicados de primer orden) que, junto al conocimiento que se tenga del dominio, tenga como consecuencia lógica el conjunto de entrenamiento del sistema. La programación lógica inductiva se define en [22] como la intersección entre el aprendizaje inductivo y la programación lógica. Esto es así porque utiliza técnicas de ambos campos:

- Del aprendizaje inductivo en los ordenadores hereda su objetivo: desarrollar herramientas y técnicas para inducir hipótesis a partir de observaciones (ejemplos) y sintetizar nuevo conocimiento a partir de la experiencia.

- De la programación lógica hereda el formalismo de representación y su orientación a la semántica. La ILP utiliza la programación lógica como mecanismo para representar las hipótesis y las observaciones, superando así dos de las principales limitaciones de las técnicas clásicas de aprendizaje en ordenadores: la rigidez en la representación del conocimiento (lógica de proposiciones, árboles de decisión, etc.) y la dificultad para expresar conocimiento de base (background knowledge) [6].

Sistemas *Data Warehousing* y Data Marts.

Data Warehouse.

Introducción.

Los tradicionales sistemas de gestión de bases de datos permiten el almacenamiento, modificación y navegación en tiempo real de los datos almacenados. Mediante un lenguaje de manipulación de datos se realizan de forma sencilla operaciones de consulta y actualización, que permiten al usuario experto inducir reglas interesantes (*"el número de ventas en la región norte creció linealmente en los últimos 5 años"*). Sin embargo, es siempre el usuario el que decide qué operaciones realizar en cada momento y el que induce la regla ante la respuesta obtenida.

Para automatizar este proceso surgen los sistemas ***Data Warehousing***, bases de datos orientadas a objetos que proporcionan nuevas posibilidades para el análisis inteligente y la inducción de conocimiento y que actualmente son el centro de la arquitectura de los Sistemas de Información de la actualidad y el entorno bajo el cual se ejecutan las técnicas especificas de la Minería de Datos [23].

Los beneficios obtenidos por la utilización de este tipo de sistemas se basan en el acceso interactivo e inmediato a información estratégica de un área de negocios. Este acercamiento de la información al usuario final permite una toma de decisiones rápida, basada en datos objetivos obtenidos a partir de bases de datos eventualmente heterogéneas de la empresa. Estos beneficios aumentan cuanto más importantes son las decisiones a tomar y cuanto más crítico es el factor tiempo. En resumen, incluyen las siguientes funcionalidades:

Integración de bases de datos heterogéneas: relacionales, documentales, geográficas, archivos de texto, etc.

Control de calidad de los datos: Aseguran no solo la consistencia e integridad de la base de datos, sino que determinan la relevancia de los datos sobre los cuales se toman las decisiones.

Análisis multidimensional (variable tiempo, etc.) y potente visualización interactiva en diferentes niveles gráficos de agrupamiento.

Un sistema ***Data Warehousing*** puede identificarse como una colección de datos orientada a sujetos, integrada, variante en el tiempo, no volátil y que soporta el proceso de toma de decisiones, brindando una sólida plataforma de datos históricos e integrados. A continuación se ven con detalle cada uno de estos términos:

Orientación a sujetos.
En compañías comerciales el mundo operacional esta diseñado alrededor de aplicaciones y funciones, como por ejemplo pagos, ventas, entregas de mercadería. Un ***Data Warehousing*** esta organizado alrededor de los mayores sujetos, como cliente, vendedor, producto y actividades El mundo operacional concierne al diseño de la base de datos y al diseño de procesos.

Un sistema ***Data Warehousing*** está enfocado exclusivamente al modelado y diseño de la base de datos. El diseño de procesos no es parte del ***Data Warehousing***.

Integración de datos.
Cuando los datos son movidos del ambiente operacional, son integrados antes de entrar en el ***Warehouse***. Valores discretos pueden representarse de formas muy distintas según el diseñador: cadenas ("rojo", "verde", etc.), caracteres ("R", "V", etc.), enumerados (0,1, etc.), etc. Independientemente de la fuente de los datos se almacenan de forma consistente: son integrados.

Variación en el tiempo.
Los datos son precisos en un momento concreto y no siempre, por lo que se consideran variantes en el tiempo. Toda estructura clave en un sistema ***Data Warehouse*** contiene de forma implícita o explícita una variable temporal. Esto no necesariamente pasa en el ambiente operacional, donde por cuestiones de rendimiento se manejan intervalos de tiempo pequeños. A su vez los datos de un sistema ***Data Warehouse***, una vez almacenados, no pueden ser modificados. En el ambiente operacional, los datos, precisos al momento de acceso, pueden ser actualizados, según sea necesario.

Arquitectura y Funcionalidad.

La figura 15 muestra la arquitectura lógica de un sistema ***Data Warehousing,*** organizada en tres niveles: las bases de datos fuentes (producción e históricos), el ***Data Warehouse*** (una base de datos con datos resumidos que son extraídos de las bases de producción) y los interfaces de usuario para la toma de decisiones.

Principalmente, el objetivo de un ambiente ***Data Warehousing*** es convertir los datos de aplicaciones del ambiente transaccional (OLTP), en datos integrados de gran calidad almacenados en estructuras optimizadas para un ambiente decisional (OLAP). Durante este proceso, los datos son transferidos a una base periódica apropiada al tipo de análisis de negocios necesario. Para ello la infraestructura funcional que lo soporta se estructura en cinco grandes grupos:

- **Acceso a Fuentes, Carga y Almacenamiento:** migración de los datos operacionales al ***Warehouse***
- **Consultas:** acceso y análisis de los datos para toma de decisiones**.**
- **Meta Datos:** base para las funcionalidades anteriores, ya que provee los datos que controlan sus procesos e interacciones.

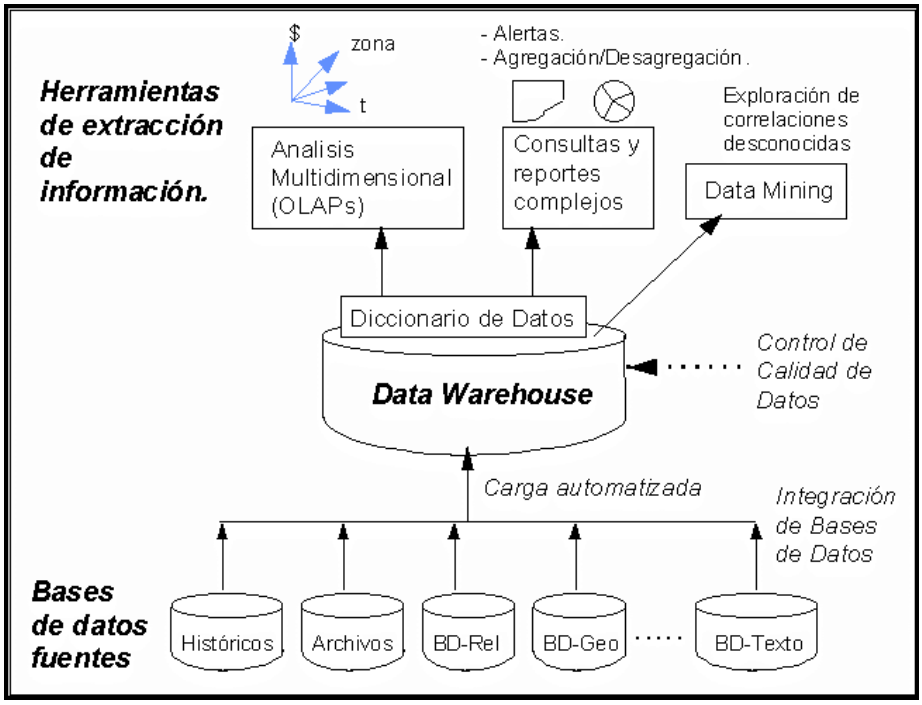


Figura 15. Estructura de un sistema ***Data Warehouse***.

Acceso a Fuentes.

Las bases de datos fuentes son típicamente las bases de datos operacionales de la organización; sin embargo, se están integrando cada vez más, a bases de distribución pública sobre industria, demografía y clientes potenciales. Los datos pueden provenir de fuentes muy diversa y determinar la mejor fuente de datos, evitando redundancias, es una de las tareas más largas y difíciles. Muchos de los procesos asociados con la función de acceso a fuentes, como **mapeo, integración, análisis y calidad de los datos**, ocurren durante la fase de análisis y diseño del ***Data Warehouse***. En realidad entre un 75 y 80% del tiempo de desarrollo del ***Warehouse*** está destinado a estas actividades.

Automatizar estas tareas, no es nada fácil. Algunas herramientas pueden ayudar a detectar problemas en la calidad de los datos, y generar programas de extracción; pero la mayor parte de la información requerida para el desarrollo, está en la mente de los analistas que trabajan con las bases de datos fuentes. Los factores que impactan directamente sobre el tiempo destinado a estas actividades son: el número de aplicativos fuentes que serán mapeados al ***Data Warehouse***, la calidad de los metadatos mantenidos en esas aplicaciones, y las reglas de empresa que las gobiernan.

Carga.

La carga comprende los procesos asociados con la migración de los datos desde los aplicativos fuentes a las bases del ***Warehouse***. Incluyen **extracción, limpieza, transformación y carga de datos**.

La **extracción** involucra acceder a los datos de los aplicativos, por lo que es necesaria una preparación previa de los datos. Hay varias alternativas de extracción que balancean el *performance* y las restricciones de tiempo y almacenamiento. Si las aplicaciones fuentes mantienen una base de datos en línea, se puede hacer una consulta que cree directamente los archivos de extracción. Debe asegurarse que no se actualicen los datos mientras se hace la extracción para no generar inconsistencias.

Por otro lado el *performance* puede caer si las transacciones en línea compiten con la extracción. Una solución alternativa es crear una vista, desde la cual extraer los datos. El inconveniente aquí, es el espacio de disco adicional para guardar esa copia de la base. El tiempo es un factor crucial; muchos aplicativos de extracción tienen un ciclo batch, en el cual transacciones fuera de línea son aplicadas a la base de datos.

La **limpieza** de los datos puede ser manejada de muchas maneras. Si los errores son inherentes a los aplicativos fuentes, los datos pueden ser limpiados sistemáticamente como parte del proceso de transformación. Desgraciadamente, muchos errores ocurren porque los aplicativos fuentes sólo tienen una mínima validación de dominio, que permite la aparición de datos inválidos. La única manera de solucionarlos es corriendo rutinas pesadas de validación a nivel de fuentes. Los errores que surgen de tipeos incorrectos, son muy difíciles de detectar y corregir.

La carga se completa con la **transformación**. Este proceso invoca reglas de conversión de valores de aplicativos locales a valores globales integrados.

Almacenamiento.

Comprende la arquitectura necesaria para integrar las vistas varias. Aunque a menudo se habla del ***Warehouse*** como si fuera un único almacén de datos, sus datos pueden estar distribuidos en múltiples bases manejadas por diferentes DBMSs. Dos tipos de gestores se ajustan bien a esta tarea: relacionales (RDBMSs) y multidimensionales (MDDBMSs). Un MDDBMS organiza los datos en un array de n dimensiones. Cada dimensión representa algún aspecto de los negocios a ser analizado.

Las bases multidimensionales presentan los datos de manera que los usuarios puedan entenderlos y accederlos fácilmente. Cada área de la empresa puede necesitar que su propia visión de los negocios sea organizada como un array multidimensional, de manera de optimizar sus requerimientos específicos. Generalmente no es deseable que la misma base multidimensional soporte los requerimientos de todas las áreas de la empresa. Una RDBMS usualmente se ajusta más, al manejo de la base integrada.

Mientras que las vistas multidimensionales son diseñadas para optimizar el acceso de usuarios finales de cada área, la base de datos integrada del *Warehouse* es diseñada para optimizar el acceso de todas las áreas.

Se le llama ***Data Warehouse*** a la base integrada, y Data Marts a las vistas multidimensionales de cada área. La separación entre el ***Data Warehouse*** corporativo y sus Data Marts satélites, introduce la necesidad de una estrategia que coordine la distribución de los datos hacia los Data Marts. Se debe considerar la incorporación de un servidor de replicación, que entregue los datos correctos, al Data Mart correcto en el momento correcto.

Los datos son almacenados en varios niveles. Los más actuales se guardan en un medio de fácil acceso en línea. Datos más viejos se pueden guardar en un medio seguro, pero más barato. Y los datos históricos pueden ser guardados en otros medios, o eliminados si ya no tienen más valor decisional.

Consultas.

Uno de los principales propósitos de las tecnologías de la Minería de Datos es chequear la efectividad de las reglas de empresa. Las herramientas de **simulación de negocios** crean modelos para testear el impacto de cambios en el ambiente de negocios. Se pueden establecer nuevas reglas de empresa. Luego hay que realimentar los aplicativos operacionales.

El arquitecto del ***Data Warehouse***, debe determinar como totalizar los datos. Existen varios enfoques viables: la sumarización puede ser hecha durante la carga, y almacenada en el ***Data Warehouse***; durante la replica a los Data Marts; o a demanda, por las herramientas de consulta y simulación.

Meta Datos.

Los Meta Datos incluyen el dominio, las reglas de validación, la derivación y la transformación de los datos extraídos. También describen las bases de datos del ***Warehouse***, incluyendo la reglas de distribución y control de la migración hacia los Data Marts. Los procesos que monitorizan los procesos del ***Warehouse*** (como extracción, carga, y uso) crean meta datos que son usados para determinar que tan bien se comporta el sistema.

Los meta datos, deberían estar disponibles para los usuarios, para ser usados en sus análisis. Los administradores pueden manejar y proveer el acceso a través de los servicios del repositorio.

El conocimiento de los metadatos es tan esencial como el conocimiento de los datos del *Data Warehouse*.

Data Marts.

Introducción.

Los Data Marts nacen para satisfacer las comunes y distintas necesidades de las distintas áreas (divisiones geográficas, departamentos corporativos, grupos de usuario, etc.) que componen la estructura de las grandes corporaciones, donde determinados recursos deben compartirse sin provocar conflictos mientras otros, por el contrario, deben ser exclusivos. Situaciones generadas por complejos sistemas ***Data Warehousing***, de difícil implantación para los técnicos y difícil navegación para los usuarios, no han tardado en encontrar en los Data Marts una respuesta con gran efectividad.

Crecimiento.

Los Data Marts nacen en principio para tratar subconjuntos del ***Data Warehouse*** (integrando también un número de fuentes heterogéneas), sin embargo actualmente están llegando a tener tamaños semejantes a los ***Data Warehouse*** corporativos de menor escala, y ello ha generado nuevos problemas de navegación:

- ***Se pierde performance a medida que aumenta el tamaño de los Data Marts***. Los usuarios esperan mejor respuesta de los Data Marts, que de los ***Data Warehouse***s.
- ***Los usuarios requieren acceso a datos de muchos Data Marts***. Los datos pueden ser replicados entre los Data Marts, pero se requieren mejores soluciones.
- ***Las compañías no pueden administrar fácilmente muchos Data Marts***. Mientras sólo se tiene un ***Data Warehouse***, se pueden tener gran número de Data Marts.
- ***Las organizaciones tienen dificultades para construir los Data Marts***. Aunque es aceptable que la construcción de un ***Data Warehouse*** lleve varios años, los Data Marts requieren un ciclo de desarrollo muy corto, para una inversión moderada.

Rendimiento.

A la hora de medir el rendimiento de un sistema ***Data Warehousing*** deben considerarse cuatro factores fundamentales: tiempo de respuesta a usuario final, datos precalculados frente a sumarización a demanda, performance en la carga y tamaño:

Red Brike ***Warehouse*** *5.0* (Red Brike Systems Inc.) posee índices que se adaptan de forma continua a los datos que están siendo procesados. Un nuevo tipo de *Join, Hash Híbrido*, maneja más eficientemente algunas situaciones como recursión en *Subjoins*.

Essbase (Arbol Software Corp.) es una bases de datos multidimensional (MDDB) que soporta actualización incremental, de manera que la estructura entera no necesita ser cambiada para cada actualización.

Pilot Decision Support Suite (de Pilot Software Inc.) provee dimensiones dinámicas, que permiten que las agregaciones sean calculadas a demanda, en lugar de preagregadas y almacenadas en el cubo. Esto puede reducir signifacativamente el tamaño del cubo, al igual que el tiempo de consolidación requerido al generar el cubo.

Data Marts Virtuales y Meta Vistas.

Los Data Marts Virtuales son vistas, de varios Data Marts físicos o del ***Data Warehouse*** corporativo, brindadas a grupos específicos de usuarios y evitando así replicas excesivas de los datos.

Sagent Data Mart Solution (Sagent Technology Inc.) provee los conceptos de **Vista Básica y Meta Vistas**. Una Vista Básica es una representación gráfica de una base de datos que incluye tablas y consultas almacenadas. Una vez que una Vista Básica es creada, múltiples Meta Vistas pueden derivar de ella. Una Meta Vista es una representación lógica de partes de una o más Vistas Básicas.

Administración de los Data Marts.

A medida que el número de Data Marts va creciendo, crece también la necesidad de administración y coordinación central, que debe garantizar permanentemente: **la consistencia e integridad en los datos (carga, almacenamiento, consultas, replicas, versiones, backup y recuperación) y los metadatos, la seguridad del sistema y la *performance* global.**

La administración debe tener un enfoque centralizado y no ser realizada entre los diferentes grupos de usuario del ***Data Warehousing***.

Alternativas.

Existen formas muy usuales en dar a los usuarios un acceso a datos de soporte decisional:

- Construir un ***Data Warehouse*** corporativo, que puede ser usado directamente por los usuarios, o puede alimentar Data Marts.
- Construir Data Marts planeados para eventualmente integrarlos en un ***Data Warehouse***.
- Construir la infraestructura para un ***Data Warehouse***, mientras al mismo tiempo, se construyen uno o más Data Marts para satisfacer las necesidades más inmediatas. Es la técnica más difundida.

La inversión en tiempo y recursos para la construcción e implantación de un sistema ***Data Warehouse*** corporativo, debe compararse con los beneficios y costes de tener un Data Mart funcionando en un plazo más corto de tiempo. Las ventajas de una u otra implantación pueden resumirse:

Data Warehouse:
Los requisitos de todas las funciones de la empresa pueden ser incluidos.
Las definiciones de los datos y las reglas de empresa son consistentes.
La redundancia de los datos, es eliminada.

Data Marts:
Menor coste y esfuerzo en una implementación inicial.
Menor tiempo de desarrollo.
La experiencia de los usuarios mejora el rendimiento.
Las funciones de cada área pueden controlar su propio Data Mart.

Aunque los Data Marts pueden proveer el éxito en solucionar muchos problemas de negocios, la proliferación de Data Marts no planeados a través de la corporación, puede llevar a inconsistencias en los datos, duplicación de éstos, y a que los usuarios no puedan acceder a todos los datos necesarios. Los vendedores están llevando a cabo algunos de estos desafíos:

- Respuesta rápida, a medida que los Data Marts crecen en tamaño.
- Administración de los Data Marts de toda la organización, para asegurar la consistencia en la definición de los datos, la seguridad, y la efectiva replica de los datos.
- Implementación rápida y repetitiva, incluyendo el acceso a Internet, para proveer la capacidad de tomar decisiones de una manera más rápida, efectiva y menos costosa.

Referencias.

[1] W.J. Frawley, G. Piatetski-Shapiro, C.J. Matheus (1991). *Knowledge Discovery in Databases: An Overview*. Knowledge Discovery in Databases, G. Piatetsky-Shapiro, W. Frawley. AAAI-MIT Press, Menlo Park, California, p. 1-27.

[2] B. DePompe (1996). *There's gold in databases.* CMP Publications (Enero).

[3] T. Hoffman, K.S. Nash (1995). ***Data Mining*** *unearths customers.* Computerworld (Julio), p. 1-28.

[4] L. Connor (1996). ***Data Warehousing*** *: Mining for data*. CMP Publications (Febrero).

[5] U.M. Fayyad, G. Piatetsky-Shapiro, P Smyth (1996). *From* ***Data Mining*** *to knowledge discovery*. Advances in knowledge discovery and ***Data Mining***, (eds) U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy. AAAI Press/MIT Press, CA., p. 1-31.

[6] A.J. Gómez Flechoso (1998). *Inducción de Conocimiento con incertidumbre en Bases de Datos Relacionales Borrosas. Tesis Doctoral.* (http://www.gsi.dit.upm.es/~anto/tesis/html/). Universidad Politécnica de Madrid.

[7] R.S. Michalski (1987). *Concept Learning*. Encyclopedia of Artificial Intelligence, Stuart C. Shapiro, Ed. John Wiley & Sons, p. 185-194.

[8] J.R. Quinlan (1986). *Induction of Decision Trees*. Machine Learning, vol. 1, p. 81-106.

[9] J.R. Quinlan (1987). *Generating Production Rules From Decision Trees*. Proceedings of IJCAI-87, Milan, Italy, p. 304-307.

[10] J. R. Quinlan (1993). *C4.5: Programs for machine learning*, Morgan Kaufmann.

[11] L. Breiman, J. H. Friedman, R. A. Olshen y C. J. Stone. *Classification and regression trees*. Wadsworth International Group. Belmont, CA, (1984).

[12] R. López de Mántaras (1991). *A Distance-Based Attribute Selection Measure for Decision Tree Induction*. Machine Learning, vol. 6(1), p. 81-92.

[13] K. Murthy and S. Kasif and S. Salzberg. *A System for Induction of Oblique Decision Trees*. Journal of Artificial Intelligence Research, (1994).

[14] J. Cendrowska (1988). *PRISM: An algorithm for inducing modular rules*. Knowledge-based systems, vol. 1, p. 255-276.

[15] T.G. Dietterich, R.S. Michalski (1984). *A Comparative Review of Selected Methods for Learning from Examples*. Machine Learning: An Artificial Intelligence Approach, R.S. Michalski, J.G. Carbonell y T.M. Mitchell, Tioga Publishing, Palo Alto, California, p. 41-81.

[16] P. Clark, T. Niblett (1989). *The CN2 Induction Algorithm. Machine Learning*, vol. 3, p. 262-283.

[17] D.W. Aha, D. Kibler, M.K. Albert (1991). *Instance-Based Learning Algorithms*. Machine Learning, vol. 6, p. 37-66.

[19] R.P. Lippmann. *An Introduction to Computing with Neural Nets*. IEEE ASSP Magazine (Abril 1987), p. 4-22.

[18] D.R. Wilson and T.R. Martinez, *Improved heterogeneous distance functions,* Journal of Artificiall Intelligence Research 6 (1997), no. 1, 1-34.

[20] J.A. Freeman, D.M. Skapura (1993). *Redes Neuronales: Algoritmos, aplicaciones y técnicas de propagación.* Addison-Wesley.

[21] J.W. Shavulk, R.J. Mooney, G.G. Towell. *Symbolic and Neural Learning Algorithms: An Experimental Comparison*. Computer Sciences Technical Report 955, University of Wisconsin (August 1990), ftp://ftp.cs.wisc.edu/tech-reports/reports/90/tr955.ps.Z.

[22] S. Muggleton. *Inductive Logic Programming*. Academic Press, San Diego (CA), 1992.

[23] A. Silberschatz, M. Stonebraker, J. Ullman. *Database Research: Achievements and Opportunities Into the 21st Century*. Report of an NSF Workshop on the Future of Database Systems Research (Mayo 1995), http://db.stanford.edu/pub/ullman/logii.ps.